Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект по курсу**

**«Операционные системы»**

**Клиент-серверная система для передачи мгновенных сообщений**

Студент: Речинская Ангелина Юрьевна

Группа: М80-206Б-20

Вариант: 2

Преподаватель: Соколов Андрей Алексеевич

Дата: 29.12.2021

Оценка: 5

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Вариант 2.**

Морской бой. Общение между сервером и клиентом необходимо организовать при помощи pipe'ов. Каждый игрок должен при запуске ввести свой логин. Должна быть предоставлена возможность отправить приглашение на игру другому игроку по логину.

**Краткое описание**

Курсовой проект состоит из двух программ – сервера и клиента. Сервер координирует вошедших клиентов друг с другом. Клиент может либо выйти, либо создать игру, иначе присоединиться к игре. Просто так клиент к игре присоединиться не может – надо, чтобы создатель игры отправил ему приглашение после создания. Как только приглашение было отправлено – клиент, которому оно пришло, способен присоединиться к игре создателя. Он подает запрос серверу и сервер смотрит, есть ли для данного пользователя приглашение. Если есть – отправляет клиенту имя лобби, иначе – выводит сообщение об ошибке. Реализация самой игры находится в программе клиента. Я не буду описывать как она устроена, так как это тогда будет не краткое описание, скажу лишь, что реализованы все функции полноценного морского боя: ввод кораблей со старта, поочередные ходы и т.д. Задано большое количество проверок для любых вводов, чтобы игра была комфортной и корректной. В целом, вот и весь функционал программы.

**Реализация**

**Файл server.cpp**

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <sys/stat.h>

#include <fstream>

#include <iostream>

#include <chrono>

#include <thread>

#include <fcntl.h>

#include <vector>

#include <utility>

#include <algorithm>

#include <sstream>

using namespace std;

vector<string> v\_namer(0);

vector<string> games(0);

vector<pair<string,string>> invites(0);

int k = 0;

void n\_reader() {

while (true) {

ifstream fin("namer");

char buff[256];

while (fin.getline(buff, 50, '\n')) {

string str = string(buff);

string identificator, action, third;

istringstream iss(str);

getline(iss, identificator, ' ');

getline(iss, action, ' ');

getline(iss, third, ' ');

if (action.length() == 0) {

v\_namer.push\_back(str);

cout << v\_namer[k] << '\n';

k++;

continue;

}

else if (action == "create") {

cout << str << '\n';

ofstream request1(identificator);

request1 << "true" << '\n';

request1.close();

games.push\_back(identificator);

continue;

}

else if (action == "invite") {

cout << str << '\n';

if (find(v\_namer.begin(), v\_namer.end(), third) != v\_namer.end()) {

ofstream request2(identificator);

request2 << "true" << '\n';

request2.close();

invites.push\_back(make\_pair(identificator, third));

continue;

}

else {

ofstream request3(identificator);

request3 << "false" << '\n';

request3.close();

continue;

}

}

else if (action == "join") {

cout << str << '\n';

int typ = 0;

for(int iterator = 0; iterator < invites.size(); iterator++){

if (invites[iterator].second == identificator) {

ofstream request4(identificator);

request4 << invites[iterator].first << '\n';

invites.erase(invites.begin() + iterator);

request4.close();

typ = 1;

break;

}

}

if (typ == 0) {

ofstream request5(identificator);

request5 << "false" << '\n';

request5.close();

}

}

}

fin.close();

}

}

int main(int argc, char\* argv[]){

if ((mkfifo("namer", 0644)) == -1){

fprintf(stderr, "Невозможно создать fifo\n");

exit(0);

}

thread name\_thread(n\_reader);

name\_thread.join();

}

**Файл client.cpp**

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <sys/stat.h>

#include <fstream>

#include <iostream>

#include <chrono>

#include <thread>

#include <fcntl.h>

#include <vector>

#include <utility>

#include <algorithm>

#include <sstream>

using namespace std;

pair<int, int> finder(pair<int, int> p, int(&f)[12][12], int k) {

if (p.first == -1) {

return make\_pair(-1, -1);

}

if (f[p.first - 1][p.second] == k) {

return make\_pair(p.first - 1, p.second);

}

else if (f[p.first + 1][p.second] == k) {

return make\_pair(p.first + 1, p.second);

}

else if (f[p.first][p.second - 1] == k) {

return make\_pair(p.first, p.second - 1);

}

else if (f[p.first][p.second + 1] == k) {

return make\_pair(p.first, p.second + 1);

}

return make\_pair(-1, -1);

}

bool isnt\_correct(string a) {

if (a.length() > 3) {

return true;

}

if (a.length() == 3 && a[2] != '0' && a[1] != '1') {

return true;

}

if (a[0] != 'a' && a[0] != 'b' && a[0] != 'v' && a[0] != 'g' && a[0] != 'd' && a[0] != 'e' && a[0] != 'j'

&& a[0] != 'z' && a[0] != 'i' && a[0] != 'k') {

return true;

}

if (a[1] < '1' || a[1] > '9') {

return true;

}

return false;

}

bool radius(pair<int, int> p, int(&f)[12][12]) {

if (f[p.first-1][p.second-1] != 3 && f[p.first - 1][p.second] != 3 && f[p.first - 1][p.second + 1] != 3

&& f[p.first][p.second - 1] != 3 && f[p.first][p.second + 1] != 3 && f[p.first + 1][p.second - 1] != 3

&& f[p.first + 1][p.second] != 3 && f[p.first + 1][p.second + 1] != 3) {

return true;

}

else {

return false;

}

}

void clear\_f(pair<int, int> p, int(&f)[12][12]) {

f[p.first][p.second] = 4;

f[p.first - 1][p.second - 1] = 2;

f[p.first - 1][p.second + 1] = 2;

f[p.first + 1][p.second - 1] = 2;

f[p.first + 1][p.second + 1] = 2;

if (f[p.first + 1][p.second] == 1) { f[p.first + 1][p.second] = 2; }

if (f[p.first - 1][p.second] == 1) { f[p.first - 1][p.second] = 2; }

if (f[p.first][p.second + 1] == 1) { f[p.first][p.second + 1] = 2; }

if (f[p.first][p.second - 1] == 1) { f[p.first][p.second - 1] = 2; }

return;

}

bool controller(pair<int, int> p, int(&f)[12][12]) {

if (f[p.first][p.second] != 4) {

fprintf(stderr, "Ошибка в корабле\n");

exit(0);

}

if (!radius(p, f)) {

return false;

}

else {

f[p.first][p.second] = 0;

pair<int, int> k = finder(p, f, 4);

pair<int, int> d = finder(k, f, 4);

if (d.first == -1 && k.first == -1) {

clear\_f(p, f);

return true;

}

else if (!radius(k, f)) {

return false;

}

else if (!radius(d, f)) {

return false;

}

else if (d.first == -1) {

f[k.first][k.second] == 0;

clear\_f(p, f);

clear\_f(k, f);

return true;

}

else {

f[k.first][k.second] == 0;

f[d.first][d.second] == 0;

pair<int, int> m = finder(d, f, 4);

pair<int, int> n = finder(p, f, 4);

if (m.first == -1) {

clear\_f(p, f);

clear\_f(k, f);

clear\_f(m, f);

if (n.first != -1) {

clear\_f(n, f);

}

return true;

}

else {

f[m.first][m.second] == 0;

pair<int, int> r = finder(m, f, 4);

clear\_f(p, f);

clear\_f(k, f);

clear\_f(m, f);

clear\_f(r, f);

if (n.first != -1) {

clear\_f(n, f);

}

return true;

}

}

}

}

bool ship\_checker(vector<pair<int,int>> v, int(&f)[12][12]) {

for (int i = 0;i < v.size() - 1;i++) {

if (v[i + 1].first != v[i].first + 1 && v[i + 1].first != v[i].first - 1 && v[i + 1].second != v[i].second + 1 && v[i + 1].second != v[i].second - 1) {

return false;

}

}

for (int i1 = 0; i1 < v.size(); i1++) {

if (!radius(v[i1], f)) {

return false;

}

for (int j = 0;j < v.size();j++) {

if (v[j].first + 1 == v[i1].first && v[j].second + 1 == v[i1].second) {

return false;

}

else if (v[j].first - 1 == v[i1].first && v[j].second + 1 == v[i1].second) {

return false;

}

else if (v[j].first - 1 == v[i1].first && v[j].second - 1 == v[i1].second) {

return false;

}

else if (v[j].first + 1 == v[i1].first && v[j].second - 1 == v[i1].second) {

return false;

}

}

}

return true;

}

void printer(int(&a)[12][12]) {

cout << " | a b v g d e j z i k |" << '\n';

cout << "--|---------------------|" << '\n';

for (int i = 1;i < 11;i++) {

if (i == 10) {

cout << i << "| ";

}

else {

cout << " " << i << "| ";

}

for (int j = 1;j < 11;j++) {

if (a[i][j] == 1) {

cout << "~ ";

}

else if (a[i][j] == 2) {

cout << "+ ";

}

else if (a[i][j] == 3) {

cout << "O ";

}

else if (a[i][j] == 4) {

cout << "X ";

}

}

cout << "|" << '\n';

}

cout << "--|---------------------|" << '\n';

}

std::pair<int,int> coordinate(string state) {

int first, second;

if (state.length() == 3) {

second = 10;

}

else {

second = state[1] - '0';

}

if (state[0] == 'a') first = 1;

else if (state[0] == 'b') first = 2;

else if (state[0] == 'v') first = 3;

else if (state[0] == 'g') first = 4;

else if (state[0] == 'd') first = 5;

else if (state[0] == 'e') first = 6;

else if (state[0] == 'j') first = 7;

else if (state[0] == 'z') first = 8;

else if (state[0] == 'i') first = 9;

else if (state[0] == 'k') first = 10;

return make\_pair(second, first);

}

//-------------------------------------МОРСКОЙ БОЙ--------------------------------------------------------------------------

void battle(string name1, string name2, string lobby\_name) {

cout << "Вы - " << name1 << ", Ваш противник - " << name2 << '\n';

int my\_field[12][12];

int opp\_field[12][12];

for (int i = 0;i < 12;i++) {

for (int j = 0;j < 12;j++) {

opp\_field[i][j] = 1;

my\_field[i][j] = 1;

}

}

pair<int, int> cord;

cout << "Вводим ваши корабли:" << '\n';

cout << "Введенные координаты должны иметь вид 'a7', буквы вводятся транслитом, ниже примеры" << '\n';

cout << "а - a, б - b, в - v, г - g, д - d, е - e, ж - j, з - z, и - i, к - k" << '\n';

cout << "Введите координаты четырехпалубника через пробел в строчку" << '\n';

string a1, b1, c1, d1;

while (cin >> a1 >> b1 >> c1 >> d1) {

if (isnt\_correct(a1) || isnt\_correct(b1) || isnt\_correct(c1) || isnt\_correct(d1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(a1);

temp.push\_back(cord);

cord = coordinate(b1);

temp.push\_back(cord);

cord = coordinate(c1);

temp.push\_back(cord);

cord = coordinate(d1);

temp.push\_back(cord);

if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

cout << "Введите координаты двух трехпалубников через пробел в две строчки" << '\n';

while (cin >> a1 >> b1 >> c1) {

if (isnt\_correct(a1) || isnt\_correct(b1) || isnt\_correct(c1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(a1);

temp.push\_back(cord);

cord = coordinate(b1);

temp.push\_back(cord);

cord = coordinate(c1);

temp.push\_back(cord);

if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

while (cin >> a1 >> b1 >> c1) {

if (isnt\_correct(a1) || isnt\_correct(b1) || isnt\_correct(c1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(a1);

temp.push\_back(cord);

cord = coordinate(b1);

temp.push\_back(cord);

cord = coordinate(c1);

temp.push\_back(cord);

if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

std::cout << "Введите координаты трех двухпалубников через пробел в три строчки" << '\n';

while (cin >> a1 >> b1) {

if (isnt\_correct(a1) || isnt\_correct(b1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(a1);

temp.push\_back(cord);

cord = coordinate(b1);

temp.push\_back(cord);

if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

while (cin >> a1 >> b1) {

if (isnt\_correct(a1) || isnt\_correct(b1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(a1);

temp.push\_back(cord);

cord = coordinate(b1);

temp.push\_back(cord);

if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

while (cin >> a1 >> b1) {

if (isnt\_correct(a1) || isnt\_correct(b1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(a1);

temp.push\_back(cord);

cord = coordinate(b1);

temp.push\_back(cord);

if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

cout << "Введите координаты четырех однопалубников в 4 строчки" << '\n';

while (cin >> a1) {

if (isnt\_correct(a1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(a1);

temp.push\_back(cord);

if (my\_field[cord.first][cord.second] == 3) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

else if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

while (cin >> b1) {

if (isnt\_correct(b1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(b1);

temp.push\_back(cord);

if (my\_field[cord.first][cord.second] == 3) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

else if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

while (cin >> c1) {

if (isnt\_correct(c1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(c1);

temp.push\_back(cord);

if (my\_field[cord.first][cord.second] == 3) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

else if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

while (cin >> d1) {

if (isnt\_correct(d1)) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

continue;

}

vector<pair<int, int>> temp;

cord = coordinate(d1);

temp.push\_back(cord);

if (my\_field[cord.first][cord.second] == 3) {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

else if (ship\_checker(temp, my\_field)) {

for (int sega = 0; sega < temp.size(); sega++) {

my\_field[temp[sega].first][temp[sega].second] = 3;

}

break;

}

else {

cout << "Некорретный корабль, попробуйте снова" << '\n';

}

}

cout << "Ваше поле: " << '\n';

printer(my\_field);

//-----------------------------------НАЧАЛО ИГРЫ----------------------------------------------------------

bool locker = true;

if (lobby\_name == name1 + "1") {

cout << "Ваш ход первый, как создателя лобби" << '\n';

}

else {

locker = false;

cout << "Вы ходите вторым, ожидайте" << '\n';

}

int my\_ships = 20, opp\_ships = 20;

while (my\_ships > 0) {

if (opp\_ships == 0) {

break;

}

if (locker) {

while (true) {

int choice;

cout << "Выберите дейтсвие: " << '\n';

cout << "1. Посмотреть свое поле" << '\n';

cout << "2. Посмотреть поле оппонента" << '\n';

cout << "3. Атаковать" << '\n';

cout << "4. Сдаться" << '\n';

cin >> choice;

if(choice == 1) {

printer(my\_field);

continue;

}

else if (choice == 2) {

printer(opp\_field);

continue;

}

else if (choice == 3) {

pair<int, int> point;

char ans[50];

string turn;

cout << "Введите ваш ход: ";

cin >> turn;

if (isnt\_correct(turn)) {

cout << "Некорректная атака, попробуйте снова" << '\n';

continue;

}

point = coordinate(turn);

if (opp\_field[point.first][point.second] == 2 || opp\_field[point.first][point.second] == 4) {

cout << "Вы уже стреляли в эту точку" << '\n';

continue;

}

ofstream gateway(lobby\_name);

gateway << turn << '\n';

gateway.close();

cout << "Заслал" << '\n';

ifstream answer(lobby\_name);

while (answer.getline(ans, 50, '\n')) {

string massage2 = string(ans);

if (massage2 == "false") {

cout << "Промах" << '\n';

opp\_field[point.first][point.second] = 2;

locker = false;

break;

}

else if (massage2 == "true") {

cout << "Попадание!" << '\n';

opp\_ships--;

opp\_field[point.first][point.second] = 4;

continue;

}

else if (massage2 == "kill") {

cout << "Корабль уничтожен!" << '\n';

opp\_field[point.first][point.second] = 4;

if (controller(point, opp\_field)) {

cout << "Успешно изменено поле" << '\n';

printer(opp\_field);

}

opp\_ships--;

continue;

}

}

break;

}

else if (choice == 4) {

ofstream gateway(lobby\_name);

gateway << "surrender" << '\n';

gateway.close();

cout << "Заслал" << '\n';

my\_ships = 0;

break;

}

}

}

else {

int commander = 0;

char ft[50];

pair<int, int> pin;

ifstream fate(lobby\_name);

while (fate.getline(ft, 50, '\n')) {

string massage = string(ft);

cout << massage << '\n';

if (massage == "surrender") {

cout << "ПОБЕДА!!" << '\n';

return;

}

else if(massage.length() != 0){

pin = coordinate(massage);

int c = my\_field[pin.first][pin.second];

if(c == 1){

commander = 1;

break;

}

if(c == 3) {

commander = 3;

break;

}

}

}

fate.close();

ofstream fate2(lobby\_name);

if (commander == 0) {

fprintf(stderr, "Странности с циклом\n");

exit(0);

}

else if (commander == 1) {

my\_field[pin.first][pin.second] = 2;

fate2 << "false" << '\n';

locker = true;

}

else if (commander == 3) {

my\_field[pin.first][pin.second] = 4;

my\_ships--;

if (controller(pin, my\_field)) {

fate2 << "kill" << '\n';

}

else {

fate2 << "true" << '\n';

}

}

fate2.close();

}

}

if (opp\_ships == 0) {

cout << "ПОБЕДА!!" << '\n';

}

else {

cout << "Пам пам пам пам... ПОРАЖЕНИЕ!!" << '\n';

}

}

//------------------------------------------------ОЖИДАНИЕ СОПЕРНИКА, СОЗДАНИЕ ЛОББИ-------------------------------------------------------------------------

void challenger(string name1) {

char buffer[50];

ofstream inv("namer");

inv << name1 << " create" << '\n';

inv.close();

ifstream my(name1);

while (my.getline(buffer, 50, '\n')) {

string massage = string(buffer);

if (massage == "true") {

cout << "Получено разрешение на создание сервера" << '\n';

break;

}

}

my.close();

string lobby\_name = name1 + "1";

char local[256];

strcpy(local, lobby\_name.c\_str());

unlink(local);

if ((mkfifo(local, 0644)) == -1) {

fprintf(stderr, "Ошибка при создании игры\n");

exit(0);

}

int prolog = 1;

string opponent;

while (prolog) {

cout << "Введите имя игрока, которого вы хотите пригласить: ";

cin >> opponent;

if (opponent == name1) {

cout << "Не стоит приглашать самого себя на бой" << '\n';

continue;

}

ofstream sr("namer");

sr << name1 << " invite " << opponent << '\n';

sr.close();

char new\_buffer[50];

ifstream new\_my(name1);

while (new\_my.getline(new\_buffer, 50, '\n')) {

string massage2 = string(new\_buffer);

if (massage2 == "true") {

cout << "Приглашение отправлено успешно" << '\n';

prolog = 0;

break;

}

else if (massage2 == "false") {

cout << "Приглашение не было отправлено, пользователя нет в сети" << '\n';

break;

}

}

new\_my.close();

}

char kecker[50];

ifstream waiter(lobby\_name);

while (waiter.getline(kecker, 50, '\n')) {

cout << "Ожидаем подключения противника..." << '\n';

string massage3 = string(kecker);

string expectation = "ok " + opponent;

if (massage3 == expectation) {

cout << "Противник найден, связываемся с противником" << '\n';

break;

}

else if (massage3.length() != 0) {

fprintf(stderr, "К вам подключилось что-то непонятное, а не оппонент, до свидания\n");

exit(0);

}

}

waiter.close();

ofstream inv2(lobby\_name);

inv2 << "ok " << name1 << '\n';

inv2.close();

battle(name1, opponent, lobby\_name);

}

//---------------------------БАЗА-------------------------------------------------------------

int main(int argc, char\* argv[]){

setlocale(LC\_ALL, "Russian");

cout << "Введите имя пользователя: ";

string name;

char buff[256];

cin >> name;

strcpy(buff, name.c\_str());

if ((mkfifo(buff, 0644)) == -1) {

fprintf(stderr, "Имя занято\n");

exit(0);

}

ofstream fout("namer");

fout << name << '\n';

fout.close();

//-----------------------------------------------------------------------------------------------------------------------------------

int choice, menu = 1, controller = 1;

while (menu) {

cout << "------------------------------------" << '\n';

cout << "Выберите дейтсвие: " << '\n';

cout << "1. Создать игру" << '\n';

cout << "2. Войти в комнату по приглашению" << '\n';

cout << "3. Выход" << '\n';

cout << "------------------------------------" << '\n';

cin >> choice;

switch (choice) {

case 1: {

challenger(name);

continue;

}

case 2: {

char buff2[256];

ofstream inv("namer");

inv << name << " join" << '\n';

inv.close();

ifstream my(name);

while (my.getline(buff2, 50, '\n')) {

string massage = string(buff2);

if (massage == "false") {

cout << "А вас никто и не приглашал!" << '\n';

break;

}

else {

cout << "Налаживаем связь с противником..." << '\n';

string game\_id = massage + "1";

ofstream inv(game\_id);

inv << "ok " << name << '\n';

inv.close();

char kecker[50];

ifstream waiter(game\_id);

while (waiter.getline(kecker, 50, '\n')) {

cout << "Ожидаем подключения противника..." << '\n';

string massage3 = string(kecker);

string expectation = "ok " + massage;

if (massage3 == expectation) {

cout << "Противник найден успешно, начинаем игру, удачи!" << '\n';

break;

}

else if (massage3.length() != 0) {

fprintf(stderr, "Вы подключились к чему-то странному, а не оппоненту, до свидания\n");

exit(0);

}

}

waiter.close();

battle(name, massage, game\_id);

}

}

my.close();

continue;

}

case 3: {

menu = 0;

string lobber = name + "1";

char local\_c[256];

strcpy(local\_c, lobber.c\_str());

unlink(local\_c);

break;

}

default: {

cout << "Неверная команда" << '\n';

continue;

}

}

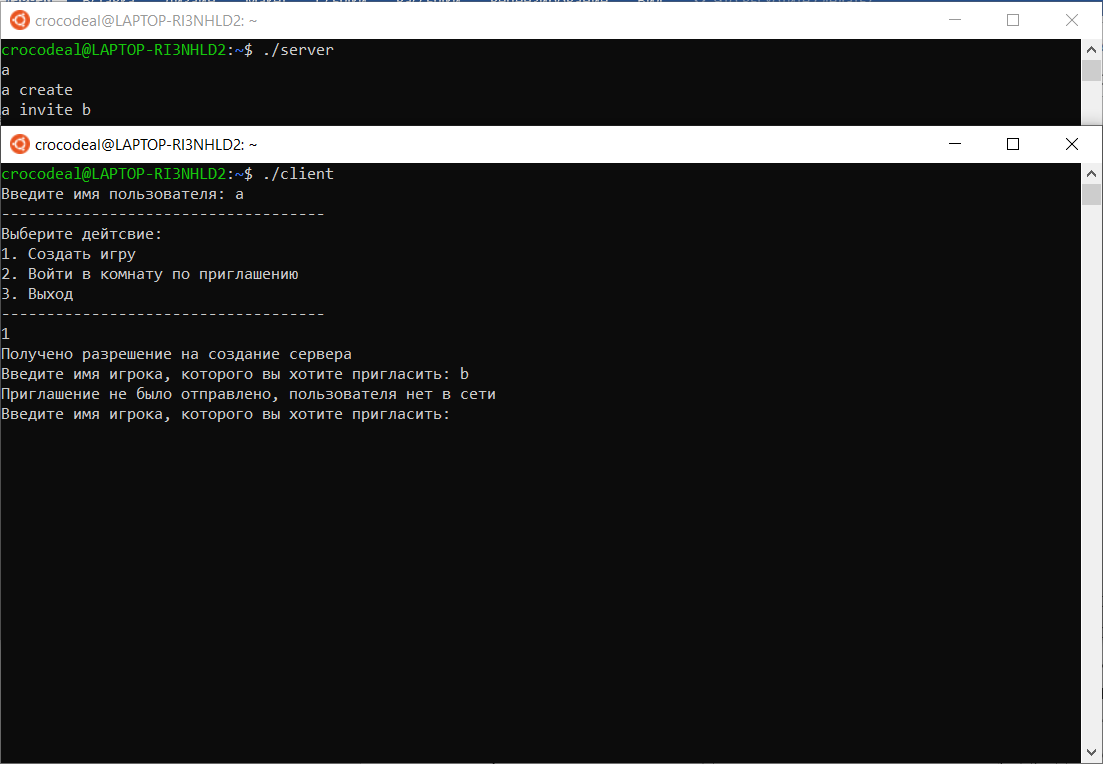
}

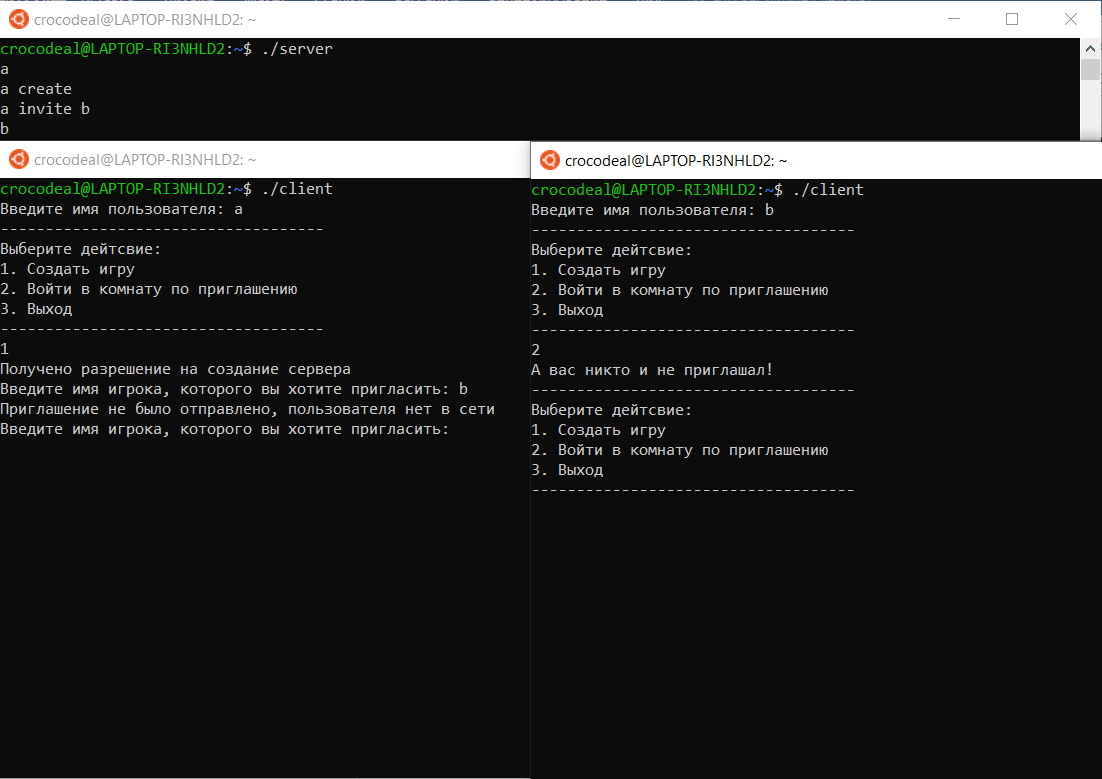
unlink(buff);

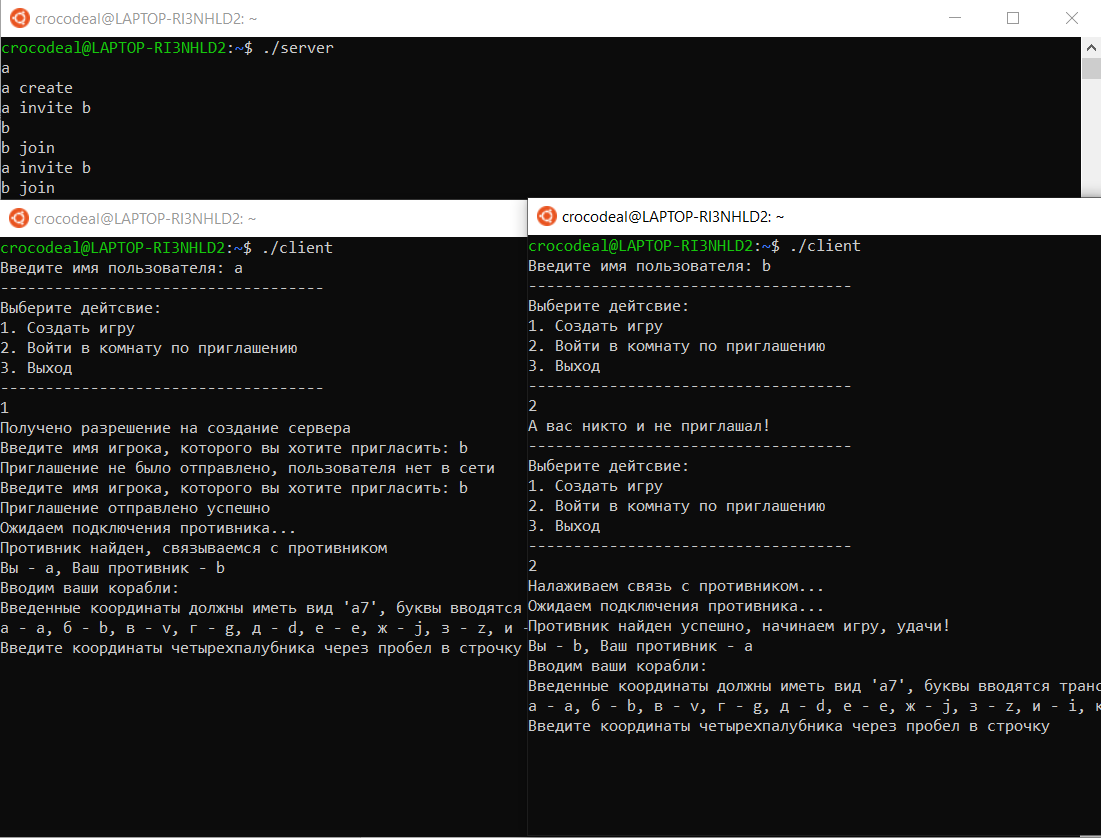
return 0;

}

**Пример работы**

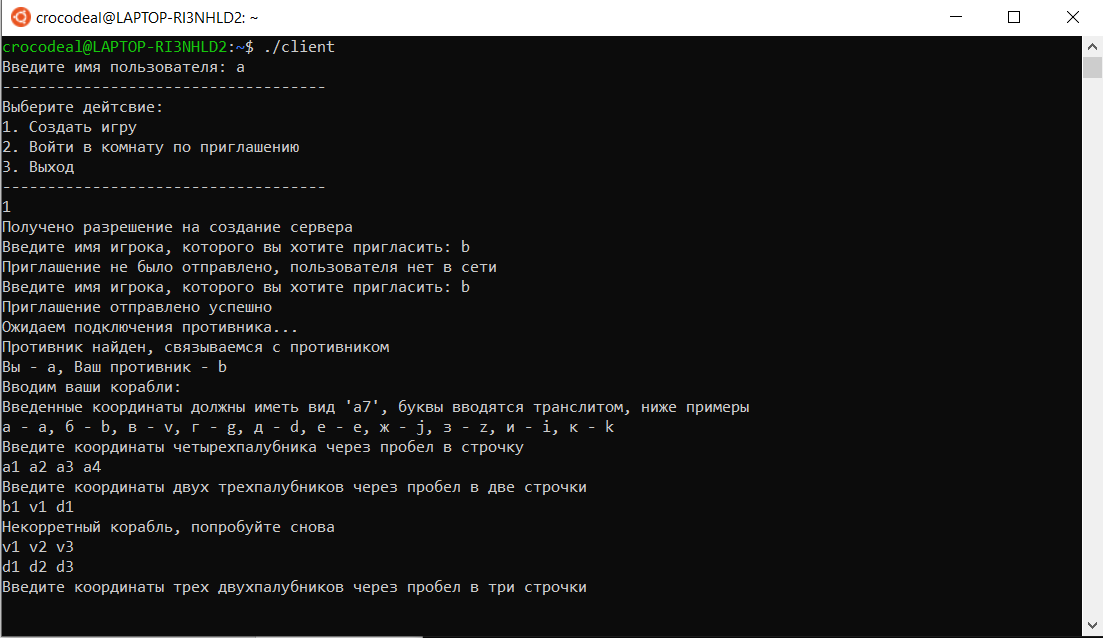




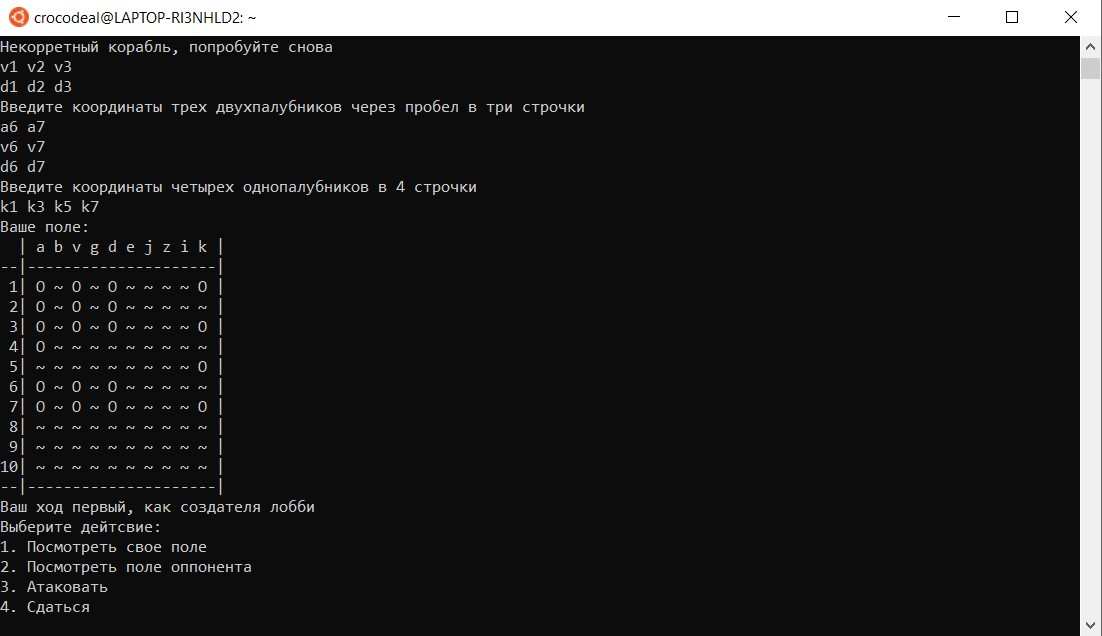


В данных скриншотах было показано последовательное взаимодействие между клиентами a и b при помощи сервера.

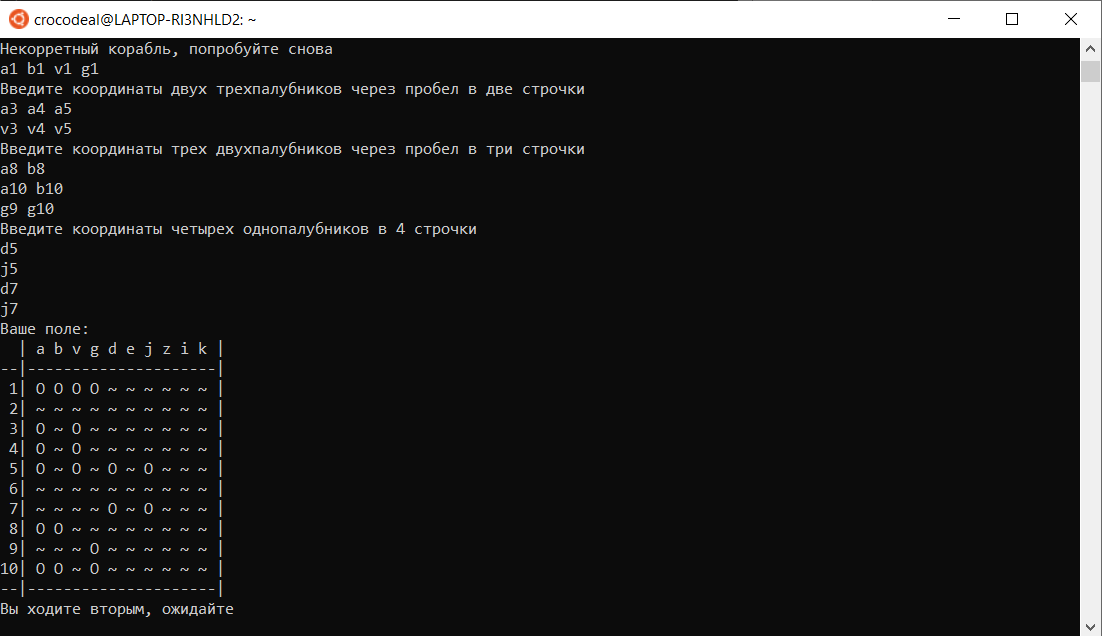
Далее будет продемонстрирована сама игра морской бой между двумя клиентами:

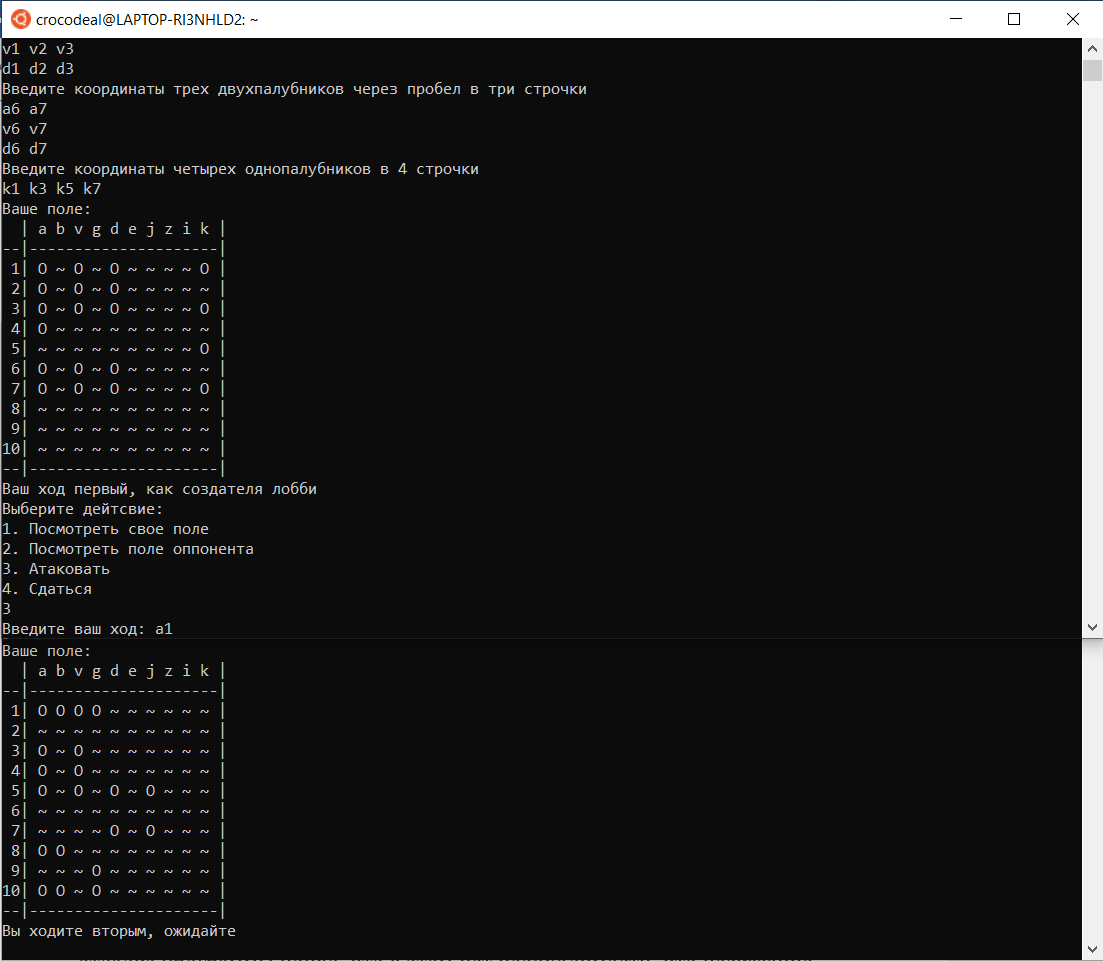


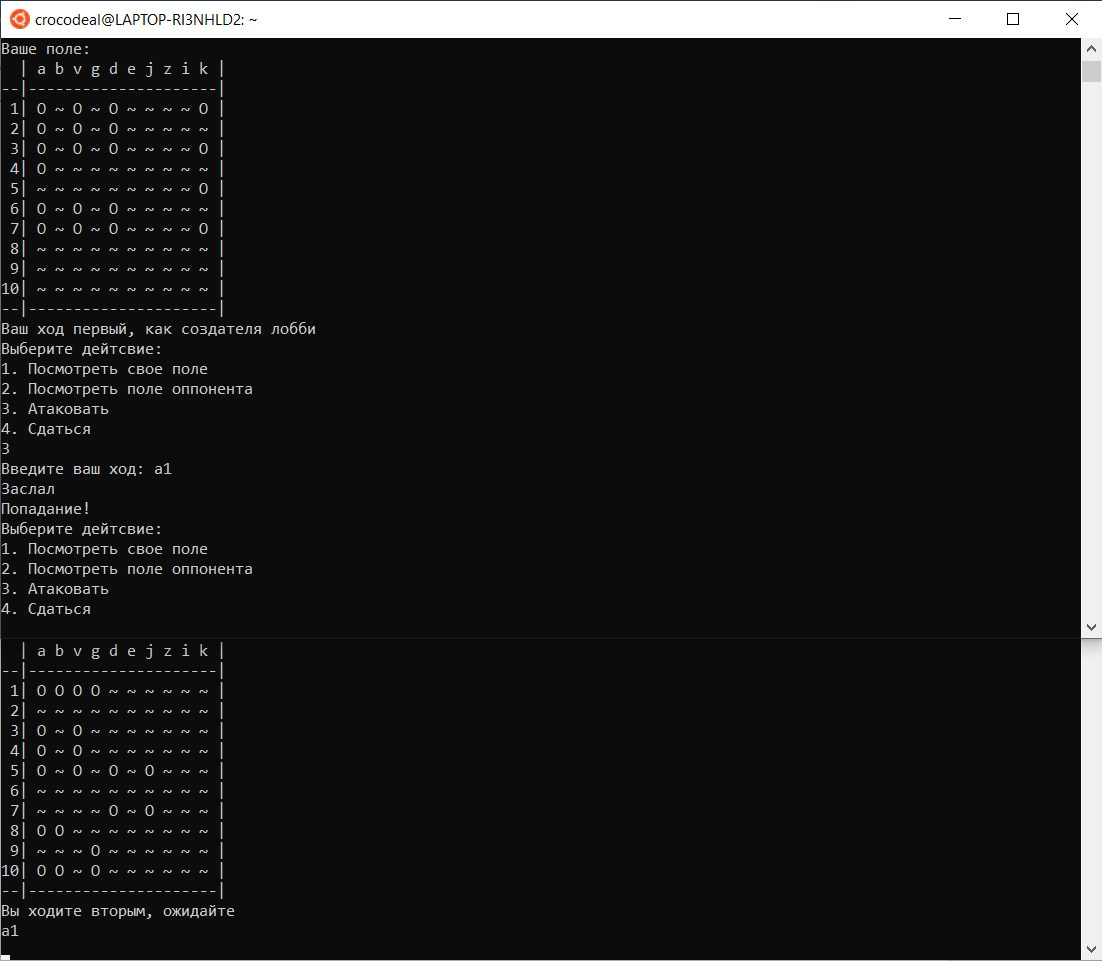
Ввод кораблей.



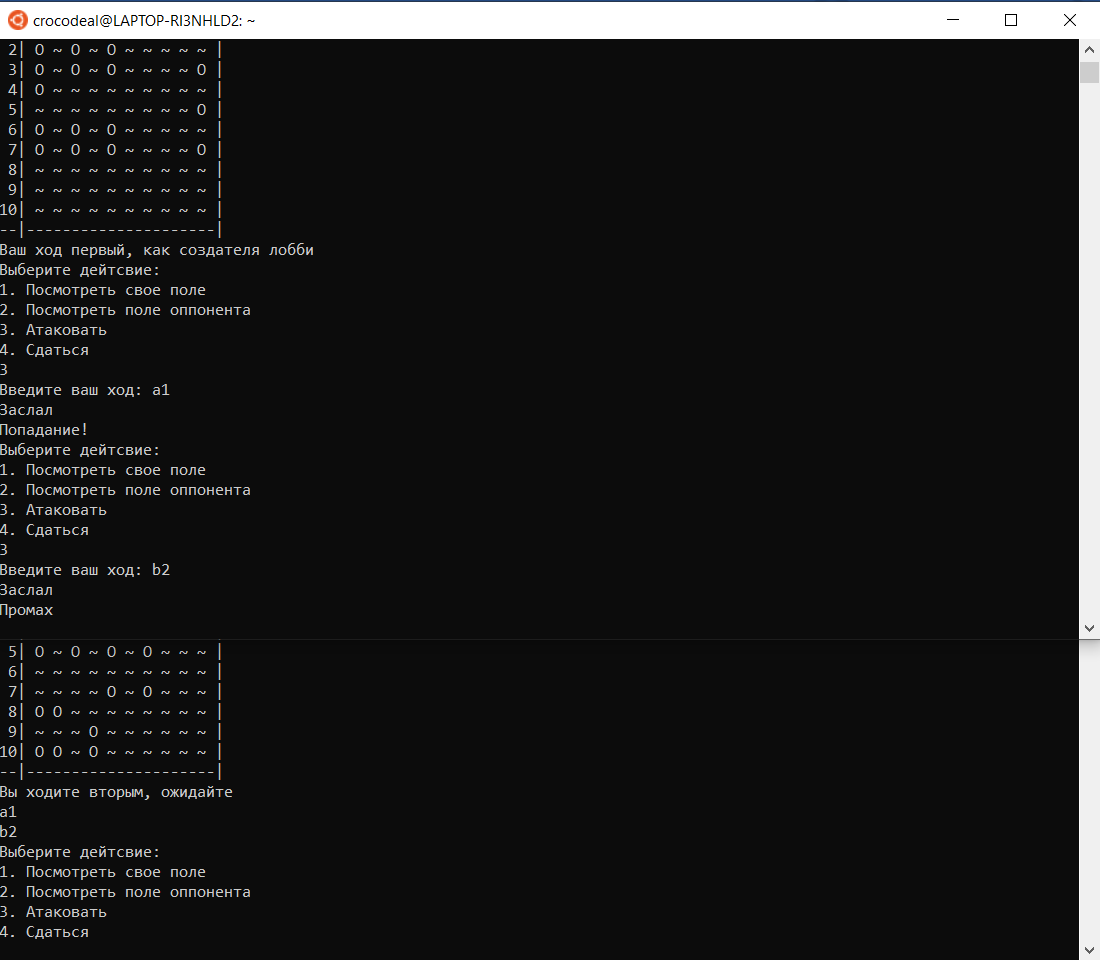
Во время вашего хода, а не хода оппонента выводится меню действий, продемонстрированное выше. Посмотрим, что же видит оппонент при атаке.



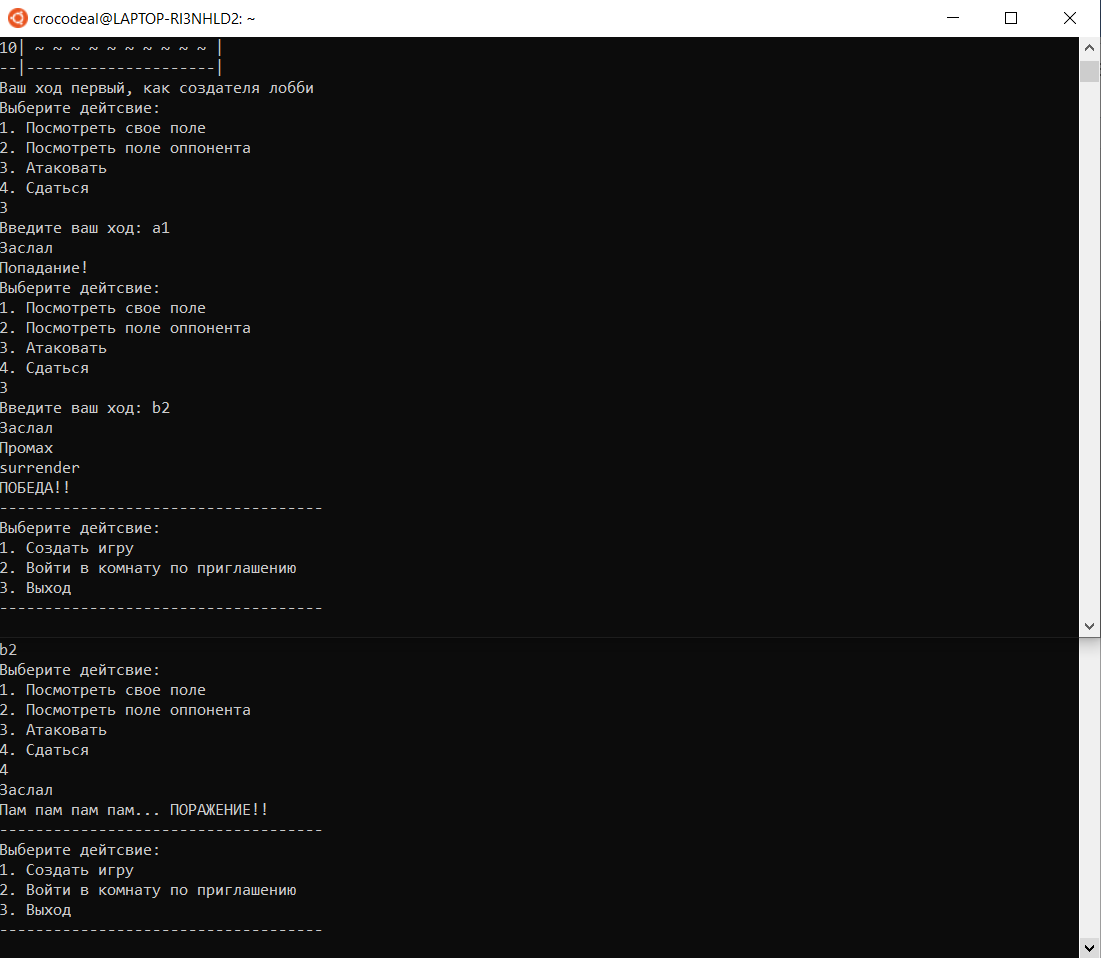




Правильно, оппонент видит ваши ходы, но ничего не может сделать с этим и ждет, пока вы промажете. Давайте промажем.



Ход передался оппоненту. Давайте завершим игру сдачей.



Видно, что игроки могут снова начать игру, если свяжутся с сервером и передадут нужные команды.

**Вывод**

Было очень интересно работать над данным курсовым проектом. В ходе работы над ним я изучила достаточно удобный способ межпроцессорного взаимодействия – именованные каналы. Мне было относительно удобно с ними работать, сначала я думала, что поочередное открытие/закрытие на чтение/запись каналов происходит с задержкой и пытался выставлять временные остановки для потоков на несколько секунд, однако при отладке программы понял, что в этом нет необходимости, что несказанно меня обрадовало и удивило в хорошем смысле этого слова. Было интересно создавать морской бой – я не думал, что такая простая игра может иметь столько тонкостей и условий при реализации на компьютере. Я потратил немало времени на придумывание корректных формулировок условий и один раз даже переписал весь код клиента из-за этого.